Summay: 2D Image Processing

Marcel A. Brusius

October 2, 2017

1 Edges and Corners

1.1 Causes of intensity changes

- boundary discontinuities
- depth discontinuities
- color/texture discontinuities
- illumination changes
- specularities
- shadows

1.2 Edge descriptors

edge direction Perpendicular to the direction of maximum intensity change

edge strength Related to the local image contrast along the normal

edge position The image position at which the edge is located

Edges correspond to extrema of derivatives.

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right] \tag{1}$$

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial x} / \frac{\partial f}{\partial y}\right) \tag{2}$$

1.3 Edges and noise

Edges are detected using finite differences

$\Rightarrow$ strong response to noise

$\Rightarrow$ use some smoothing beforehand

$\Rightarrow$ often convolution with derivative of a Gaussian kernel $f \star \frac{d}{dx} g$

1.4 Optimal edge detector

good detection	minimize probability of false positives and false negatives
good localization	detected edges have to be as close as possible to true edges
single response	detector must return one point only for each true edge point (minimize number of local maxima around true edge)

1.5 Canny edge detector

- filter image with derivative of Gaussian
- find magnitude and orientation of gradient
- non-maxima suppression
- linking and thresholding $\Rightarrow$ two thresholds (hysteresis)
  - high threshold: start edge curve
  - low threshold: continue edge curve
  - hysteresis: suppress isolated pixels, as they are more likely to be a result of noise

1.6 Corner classification

flat	no change in any direction
edge	no change along the edge direction
corner	significant change in all directions

1.7 Harris corner detector

- Error function

$$E(u,v) = \sum_{(x,y)\in W} \left[I(x+u, y+v) - I(x,y)\right]^2 \tag{3}$$

- Small motion assumption:
  - Taylor expansion
  - Ignore 2nd order terms and higher order terms
- Error function after Taylor

$$E(u,v) = \sum_{(x,y)\in W} \begin{pmatrix} u & v \end{pmatrix} \underbrace{\begin{pmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{pmatrix}}_{H} \begin{pmatrix} u \\ v \end{pmatrix} \tag{4}$$

$$I_x = \frac{\partial}{\partial x} G(x, y, \sigma_D) \cdot I(x_i, y_i) \tag{5}$$

$$I_y = \frac{\partial}{\partial y} G(x, y, \sigma_D) \cdot I(x_i, y_i) \tag{6}$$

- Differentiation scale $\sigma_D \Rightarrow$ smoothing prior to image derivative
- Integration scale $\sigma_I \Rightarrow$ integrating derivative responses (Gaussian window size)

  $\rightarrow\ \sigma_I = \gamma \sigma_D$
- Eigenvalues $\lambda_{+/-}$ of H
- Choose points with large response $\lambda_- > T$, T denotes the threshold
- Choose points where λ_- is a local maximum as features
- Harris operator

$$f = \frac{\det(H)}{\operatorname{tr}(H)} \tag{7}$$

- Corner response, with empirical constant k,

$$R = \det(H) - k \operatorname{tr}(H)^2 \tag{8}$$

  - corner: R is large
  - edge: R negative with large magnitude
  - flat: $|R|$ small
- Rotation invariant

- **PARTIALLY** invariant to linear intensity change
- **NOT** invariant to scaling
  - → if scale known: set σ_D, σ_I accordingly
  - → if scale unknown detect interest points at multiple scales
- **NOT** invariant to viewpoint change
- **NOT** invariant to contrast change
- most repeatable detector

1.8 Automatic scale selection

- design a function $F(x, \sigma_n)$ which provides a local measurement
  - should be rotation invariant
  - should have one stable sharp peak
  - Square gradient, LoG (→ yields best results), DoG, Harris function
- select points at which $F(x, \sigma_n)$ attains a maximal over σ_n
- normalize image part to a fixed size
- relation $\frac{\sigma_1}{\sigma_2}$ reveals scale factor between images

1.9 Blobs

- superposition of two ripples
- the magnitude of the Laplacian response will achieve a maximum at the center of the blob (if scale of Laplacian matches scale of blob)
- Laplacian decays as scale increases
  - → multiply Gaussian derivative by σ
  - → multiply Laplacian (2nd Gaussian derivative) by σ^2
- characteristic scale
  - scale that produces a peak of the Laplacian response in the blob center
- Approximate the Laplacian with a difference of Gaussians

$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k-1)\sigma^2 \nabla^2 G \tag{9}$$

2 Optical flow

2.1 Motion field

The motion field is the projection of the 3D scene motion into the image

2.2 Optical flow

- The apparent motion of brightness patterns in the image
- **ideally**: optical flow = motion field
- **in general** optical flow $\neq$ motion field
  - apparent motion can be caused by lightning changes without any actual motion

2.3 Feature tracking

- small motion $\Rightarrow$ easy to track
-

Challenges

- which features can be tracked
- efficient tracking across frames
- points may change appearance (moving to shadows, rotations)
- drift error: accumulated small errors due to model update
- appearing/disappearing points (add/delete)

2.4 Assumptions

- Brightness consistency
  - Brightness in small regions remains the same across frames
  - Brightness Consistancy Equation

$$I(x, y, t) = I(x + u, y + v, t + 1) \tag{10}$$
$$I(x + u, y + v, t + 1) \approx I(x, y, t) + I_x u + I_y v + I_t \tag{11}$$
$$0 \approx I_x u + I_y v + I_t \tag{12}$$
$$0 = \nabla I \cdot \begin{pmatrix} u & v \end{pmatrix}^T + I_t \tag{13}$$

- One equation, two (u, v) unknowns

$\Rightarrow$ Aperture problem

- Spatial coherence
  - neighboring points belong to the same surface $\Rightarrow$ similar motion
  - project to nearby pixels $\Rightarrow$ spatial coherence
  - use brightness consistency equation and a windows of size $w \times w$

  $\Rightarrow$ obtain w^2 equations of the form

$$0 = \nabla I(p_i) \cdot \begin{pmatrix} u & v \end{pmatrix}^T + I_t(p_i), \qquad i = 1, \ldots, w^2 \tag{14}$$

  - for RGB images: $3 \cdot w^2$ equaions
  - **Lucas-Kanade flow**: least squares problem

$$\underset{x}{\text{argmin}} \quad \|Ax - b\|_2^2 \tag{15}$$

$$x = (A^T A)^{-1} A^T b \tag{16}$$

  - conditions on $A^T A$:
    - should be invertible
    - should not be too small (due to noise, eigenvalue constraint)
    - should be well-conditioned (quotient $\frac{\lambda_1}{\lambda_2}$ should not be too large)
  - see criteria for Harris corner detector

- Temporal persistence
  - image motion of a surface patch changes gradually over time

2.5 Computing optical flow

- error in optical flow constraint

$$e_c = \int \int (I_x u + I_y v + I_t)^2 dx dy \tag{17}$$

- smoothness constraint

$$e_s = \int \int (u_x^2 + u_y^2) + (v_x^2 + v_y^2) dx dy \tag{18}$$

- penalizing error terms, find minimizer (u, v) for

$$e = e_c + \lambda e_s \tag{19}$$

2.6 Optical flow algorithm

- differentiate e with respect to $v_{i,j}$ and $u_{i,j}$, and set to zero
- define mean values $\bar{v}_{i,j}, \bar{u}_{i,j}$ locally around pixel (i, j)
- define update constant

$$\lambda^n = \frac{I_x^{i,j}\bar{u}_{i,j}^n + I_y^{i,j}\bar{v}_{i,j}^n + I_t^{i,j}}{\lambda + (I_x^{i,j})^2 + (I_y^{i,j})^2} \tag{20}$$

- optical flow update rule

$$\bar{u}_{i,j}^{n+1} = \bar{u}_{i,j}^n - \lambda^n I_x^{i,j} \tag{21}$$
$$\bar{v}_{i,j}^{n+1} = \bar{v}_{i,j}^n - \lambda^n I_y^{i,j} \tag{22}$$

2.7 Aliasing in optical flow

- temporal aliasing causes ambiguities in optical flow (many pixels with same intensity)
- remedy: coarse-to-fine estimation (reduce resolution $\Rightarrow$ image pyramide)
- image pyramide: iteratively apply Lucas-Kanade and upsample inbetween

3 Support vectors

- decision hyperplane $w^T x + b = 0$
- decision function $D(x_i) = \text{sign}(w^T x_i + b)$
- margin hyperplanes:

$$w^T x + b = \gamma \tag{23}$$
$$w^T x + b = -\gamma \tag{24}$$

- scale invariant ($\gamma \equiv 1$)
- maximize margin

$$\underset{w}{\text{argmax}} \quad \frac{2}{\|w\|} \tag{25}$$

3.1 Nonlinear SVMs

- map input space (data) into higher-dimensional feature space where the data is separable

3.2 The kernel trick

- define a kernel function K such that

$$K(x, y) = \phi(x) \cdot \phi(y) \tag{26}$$

- linear SVM decision function (α_i learnt weight, x_i support vector, y_i class label)

$$w^T x + b = \sum_i \alpha_i y_i x_i^T x + b \tag{27}$$

- kernel SVM decision function

$$\sum_i \alpha_i y_i \phi(x_i)\phi(x) + b = \sum_i \alpha_i y_i K(x_i, x) + b \tag{28}$$

- polynomial kernel $K(x, y) = (c + x^T y)^d, d \in \mathbb{N}$
- Gaussian kernel $K(x, y) = \exp(-\frac{1}{\sigma^2}\|x - y\|)$

3.3 Over- and underfitting

- Underfitting
  - model does an equally poor job on training set and test set
  - probably training procedure ineffective
  - probably model too simple to represent data
- Overfitting
  - model fits irrelevant characteristics (noise) in training data
  - probably model too complex
  - probably amount of training data insufficient

3.4 Validation

- split data into training, validation and test set
- use training to optimize model parameters
- use validation to choose best model
- use test set only to evaluate performance

4 Integral images

- compute a value for each pixel (x, y) that represents the sum of all pixels above and to the left of (x, y), inclusive
- easy preprocessing step
- evaluate ALL rectangle sizes in constant time
- NO image scaling necessary
- scale rectangular features instead

5 AdaBoost

5.1 Boosting

- classification scheme
- combine *weak* learners into a more accurate ensemble classifier
- weak learner should be better than chance
- during boosting round, select weak learner that does well on examples that were hard for previous weak learners
- Complexity: $\mathcal{O}(MNK)$, M rounds, N examples, K features

5.2 Attentional cascade

- start with simple classifier to reject many negative sub-windows while detecting almost all positive sub-windows
- positive response triggers evaluation of second more complex classifier
- negative response leads to immediate rejection of the sub-window
- chain classifiers that are progressively more complex and have lower false positive rates

5.3 Training the cascade

- set target detection and false positive rates for each stage
- keep adding features to the current stage until its target rates have been met
- if overall false positive rate NOT low enough, add another stage
- use false positives from current stage as negative training examples for next stage

5.4 Challenges

- Lighting changes
- Occlusion
- Point of view

5.5 Eigenfaces

- most face images lie in a low-dimensional subspace given by the first $k < d$ directions of maximum variance
- PCA $\Rightarrow$ determine vectors/eigenfaces $u_1, \ldots, u_k$ that span the subspace
- represent all faces in dataset as linear combination of eigenfaces

6 Bayesian Tracking

6.1 Bayesian filtering

- probability distributions represent our belief about the stage of a dynamical system
- recursive cycle, repeat:
  - predict from motion model
  - sensor measurement
  - correct prediction

6.2 Notation

- state: x_t state of the system at time t
- measurement: z_t measurement at time t
- measurement: $z_{t_1:t_2}$ measurement from time t_1 to t_2
- control data: u_t control data at time t, corresponds to the change of the state in time $]t-1, t]$
- control data sequence: $u_{t_1:t_2} = u_{t_1}, u_{t_1+1}, \ldots, u_{t_2}$
- Markov assumption:

$$p(x_t|x_{0:t-1}, z_{1:t-1}, u_{1:t}) = p(x_t|x_{t-1}, u_t) \tag{29}$$

$$p(z_t|x_{0:t}, z_{1:t-1}, u_{1:t}) = p(z_t|x_t) \tag{30}$$

6.3 Bayesian inference

- Posterior distribution after including measurement z_t: $\text{bel}(x_t) = p(x_t|z_{1:t}, u_{1:t})$
- Prediction before including measurement z_t: $\overline{\text{bel}}(x_t) = p(x_t|z_{1:t-1}, u_{1:t})$

6.4 Recursive Bayesian filtering

$$\underbrace{\text{bel}(x_t)}_{\text{posterior}} = \eta \underbrace{p(z_t|x_t)}_{\text{likelihood}} \underbrace{\int \underbrace{p(x_t|x_{t-1}, u_t)}_{\text{motion model}} \underbrace{\text{bel}(x_{t-1})}_{\text{posterior at } t-1} \, dx_{t-1}}_{\overline{\text{bel}}(x_{t-1})} \tag{31}$$

6.5 Kalman filter

- motion model: $x_t = A_t x_{t-1} + B_t u_t + \epsilon_t, \epsilon_t \sim N(0, R_t)$
- measurement model: $z_t = C_t x_t + \delta_t, \delta_t \sim N(0, Q_t)$

A_t describes how the state evolves from $t-1$ to t without control input or noise

B_t describes how the control input u_t changes the state from $t-1$ to t

C_t describes how to map the state x_t to an observation z_t

- Only predicting would lead to **diverging** paths

6.6 Extended Kalman filter

- problem

$$\underbrace{\text{bel}(x_t)}_{\text{non-Gauss}} = \eta \underbrace{p(z_t|x_t)}_{\text{nonlinear + Gauss}} \int \underbrace{p(x_t|x_{t-1}, u_t)}_{\text{nonlinear + Gauss}} \underbrace{\text{bel}(x_{t-1})}_{\text{Gauss}} dx_{t-1} \tag{32}$$

- first order Taylor expansion

$$y = f(x) \approx f(a) + \nabla_x f(a)(x - a) \tag{33}$$

6.7 Particle filter

- Initialization: randomly draw particles in state space
- Measurement update: multiply weights with measurement likelihood
- Resampling: randomly draw new particles from the old set with probabilities proportional to weights
- Time update: Move particles according to the motion model, diffuse particles by adding noise

7 Convolutional Neural Networks

- Convolutional Layer
  - input is a gray value (2 dimensional) or RGB (3 dimensional) image
  - convolution of small filter mask with a small region in the input image
  - shared weights for each pixel
  - Rectified Linear Unit (ReLU), or differentiable approximation

$$f(x) = \max(0, x) \tag{34}$$
$$g(x) = \ln(1 + \exp(x)) \tag{35}$$

  - emulation of response of individual neuron to visual stimuli
- Pooling Layer
  - discard unnecessary information
  - popular: Max-Pooling, 2×2 window, only take largest response
  - less memory consumption
  - higher processing speed
  - deeper networks possible $\Rightarrow$ solve more complex tasks
  - can prevent overfitting
  - make network robust against small shifts
- Fully-Connected Layer
  - final classification